STATE-TRANSITION COST FUNCTIONS AND AN APPLICATION TO LANGUAGE TRANSLATION

Hiyan Alshawi
Adam L. Buchsbaum
AT&T Labs, 600 Mountain Ave., Murray Hill, NJ 07974, USA.

ABSTRACT
We define a general method for ranking the solutions of a search process by associating costs with equivalence classes of state transitions of the process. We show how the method accommodates models based on probabilistic, discriminative, and distance cost functions, including assignment of costs to unseen events. By applying the method to our machine translation prototype, we are able to experiment with different cost functions and training procedures, including an unsupervised procedure for training the numerical parameters of our English-Chinese translation model. Results from these experiments show that the choice of cost function leads to significant differences in translation quality.

1 INTRODUCTION
Standard applications of preference scores in machine translation [3, 4, 15], and more generally for disambiguation in language processing [8, 14], employ special-purpose training and scoring mechanisms. To gain more flexibility in assigning costs to translations produced by a speech translation system, we abstract the counts, weights, and penalties used to score a search process and provide a general costing mechanism that is independent of the details of the algorithms and information being applied.

An earlier version of our translation prototype lacked scoring flexibility in two ways. First, the costs were always interpreted as negated log probabilities of a statistical translation model. This precluded using all the available data. In particular, it did not use examples of “incorrect” translations, nor could it treat correctness of solutions as a matter of degree.

Second, the particular statistical model for deriving translations was hard-wired into the mechanisms for counting events and computing and applying the costs. This made it inconvenient to alter the model, for example to experiment with context dependence.

To gain flexibility, we separate much of the costing mechanism into a module that is independent from its application to translation. This facilitates experimentation with variants of the translation model, and with ways of associating costs with solutions, that are different from the standard probabilistic approach.

A particular example is an unsupervised method for associating costs with actions in a translation model. The model structure is defined in advance, but the numerical costs account for a graded view of the “goodness” of a translation derivation. To do this we compute a distance metric between a source-language sentence and the result of translating it into a target-language and back again. Cost penalties associated with “irregular” events leading to incomplete solutions are handled systematically in this framework.

In Section 2 we present the framework for associating costs with actions and solutions of a search process. We briefly review our translation model in Section 3 and discuss some of the cost functions we have tried. Section 4 describes the implementation of the costs module and the training procedures, and Section 5 presents our results.

2 PROCESS EVENTS AND COSTS
2.1 Costed Non-deterministic Processes
We generalize our translation problem in terms of costed non-deterministic processes. Such a process can perform multiple input and output operations. Here, we restrict ourselves to simple costed non-deterministic processes, which read a string, execute a sequence of internal actions, write a pair consisting of a string and a real number, and then halt. A real number cost is associated with each action. The output string is a solution, and the cost of that solution is the sum of the costs for all actions prior to halting.

For translation, the input could be a sentence (or a speech signal or word lattice), the solution a possible translation, and the cost some measure of the distortion of meaning effected by this translation.

Given a characterization of a simple costed non-deterministic process, the standard search problem is to find a minimum-cost solution. It is also often of practical importance to find a solution that has a high probability of having the lowest cost, or to find an approximate solution with a cost “close” to the minimal cost. Sometimes we want the \( N \) lowest-cost solutions for some integer \( N \).

2.2 Events and Contexts
The cost of performing an action is determined by the internal state of the process before taking the action and by the action taken. For complex applications such as translation or speech recognition, we assume that the cost depends only on certain aspects of the state and the action taken.

We formalize this assumption by defining the cost of taking an action as a function of equivalence classes of states, which we call contexts, and equivalence classes of actions, which we call events. An event \( e \) in a context \( c \) forms a choice, written as \( (e|c) \).

The sets of allowable actions from states induces sets of allowable events for each context \( c \) in the obvious way. Together, the events, contexts, specification of allowable events, and cost function constitute a process cost model, or simply a model. The costs specified by the cost function are the model parameters; the other components constitute the model structure.

We contrast our costed event framework to a related effort toward providing tools for processing weighted automata [13]. The weighted-automata tools provide algorithms for manipulating labeled, weighted, state transition networks. We further separate the algorithmic issues from those relating to cost manipulation. For example, the implementation of a speech recognizer using the weighted-automata tools would be an instance of a costed process, and the costs for
arcs could be trained using our costed event model. Our framework, however, can be used in conjunction with any search process in which state transitions can be traced, regardless of the algorithms being applied.

2.3 Probabilistic Model

We create a standard probabilistic model given criteria for evaluating the utility of a solution. For example, consider some solutions to be judged positive and others negative. Given a set of solutions, let \( n^+ (e|c) \) (resp., \( n^- (e|c) \)) be the number of times choice \( (e|c) \) was taken leading to positive (resp., negative) solutions and \( n^0 (c) \) (resp., \( n^- (c) \)) be the number of times context \( c \) was encountered for these solutions. We then estimate costs as usual: \( f'(i|e|c) = \ln(n^0 (c)) - \ln(n^+ (e|c)) \). Assuming this estimate is exact, a minimum-cost solution results from a highest probability sequence of events, given that the solution is positive.

2.4 Discriminative and Distance Models

We use the discriminative model to exploit information from negative solutions. The cost function for the discriminative model is \( f'(i|e|c) = \ln(n^- (e|c)) - \ln(n^0 (e|c)) \). This estimates the negated log of the likelihood ratio comparing the probability, conditional on context \( c \), that event \( r \) is taken on a path leading to a positive solution as opposed to being taken on a path leading to a negative solution (See Dunning [7] on the application of likelihood ratios in computational linguistics.)

Both the above models judge solution quality coarsely. We have therefore experimented with another class of models that make use of an error or distance function. For the general case, we have a function \( h(s, t) \equiv d \), where \( s \) is the input, \( t \) is a solution, and \( d \) is a non-negative real number. The distance function yields greater numbers for worse solutions; it is zero for ideal solutions.

The parameters for the mean distance model [2] are \( E_0 (e|c) \), the average of \( h(s, t) \) for pairs \( s, t \) produced by a sequence of choices including \( e \) in context \( c \).

Note that the mean distance model does not take into account that particular choices faced by a process are choices between events with the same context. It is also somewhat sensitive to peculiarities of the underlying distance function. We therefore introduce the relative distance model. Let \( E_0 (e|c) \) be the average of \( h(s, t) \) for source-solution pairs \( s, t \) produced by a sequence of choices including \( e \) in context \( c \).

The cost function \( f(r|e|c) \) for the relative distance model is defined as \( f(r|e|c) = E_0 (e|c) / E_0 (c) \). A relative distance cost is the ratio of the expected distance for solutions involving an event in a particular context and the expected distance for solutions involving any allowable event for that context. Therefore, "good" events, i.e., those that tend to reduce the cost of a solution, have costs less than 1, whereas "bad" events have costs greater than 1. A cost of 1 indicates that the model is indifferent to the event, as compared to other available choices. This provides a specific neutral value as the cost for unseen events.

2.5 Unseen Event Costs

Our approach to deriving costs for unseen choices generalizes the notion of "backed-off" likelihood estimates [5, 10, 11]. (Clustering techniques [12] are an alternative.)

When faced with an unseen choice \( e|c \), we use the cost, suitably normalized, for \( (c'|c) \), where \( c' \) is a more general context than \( c \). We could also apply a similar generalization to the equivalence class for the event \( e \), but we have not yet tried this in our experiments.

There are many ways of generalizing \( c \) to \( c' \); we experimented with the following. Given a context \( c = \{ b_1, \ldots, b_m \} \), where the \( b_i \)'s are features of state, let \( c_i = \{ b_i, \ldots, b_{i+1}, b_{i+2}, \ldots, b_m \} \), for \( 1 \leq i \leq m \). We define \( c' \) as the equivalence class of contexts \( c_1, \ldots, c_m \). The process can be applied recursively.

The cost corresponding to the logical conclusion of this process, i.e., the cost for \( (c_0|c_0) \) where \( c_0 \) are unrestricted event and context equivalence classes, is the default cost for the model. In particular, for models with relative distance costs, the default cost is 1, as noted earlier.

3 APPLICATION TO TRANSLATION

3.1 Review of Translation Model

The experiments reported in Section 5 use a transfer-based translation system. Monolingual analysis and generation components analyze and generate dependency graphs like those employed by dependency grammars [9]. The graph nodes are labeled with words and the arcs with relation symbols. A transfer component maps dependency graphs with source-language words into corresponding graphs with target-language words. Each component has an associated model. Alshawi [1] describes relevant head machine models and algorithms for applying them efficiently to translation.

The choices of the analysis model provide a good illustration of the structure of a costed model. When a head move \( m \) is applied to analyze the dependents of a head word \( w_0 \), it starts with an incomplete phrase consisting only of \( w_0 \). Each state transition of \( m \) extends this phrase to the left or right by considering a dependent phrase. When \( m \) stops, the phrase is considered complete. In the following choices, \( w_1 \) is a dependent of \( w_0 \) under relation \( r \) (an \( r \)-dependent of \( w_0 \)), and \( s \) and \( s' \) are states of \( m \).

- \((m|w_0)\) Machine \( m \) may construct the dependent arcs with head \( w_0 \).
- \((s|m|w_0)\) Machine \( m \) for head word \( w_0 \) may start in state \( s \).
- \((left|s'|r|m|s)\) In state \( s \), machine \( m \) may construct an arc for a left-\( r \)-dependent and enter state \( s' \).
- \((right|s'|r|m|s)\) In state \( s \), machine \( m \) may construct an arc for a right-\( r \)-dependent and enter state \( s' \).
- \((w_1|m|w_0|s)\) An incomplete arc with label \( r \) from head \( w_0 \) may be completed by selecting a dependent subphrase headed by \( w_1 \).
- \((stop|m|s)\) Machine \( m \) may stop in state \( s \).

3.2 Translation Distance Functions

Given a translated corpus, we can define a function for a distance-based model in terms of a string distance metric \( d \). The distance function \( h \) of Section 2.4 would be \( h(s, t) = d(t, t_o) \), where \( t_o \) is the human translation of \( s \) found in the corpus. This model could exploit automatically aligned corpora of human-generated translations [3, 16].

Lacking a translated corpus, we can apply such a distance function by retranslating the target string into the source language and measuring the distance between the resulting string and the original source. We refer to this as the back-translation process and note that it is completely unsupervised. For this to be effective, the structure of the translation model (in both directions) needs to be reasonably constraining; otherwise, the parameters of the translation model will tend to prefer translations that mirror the source language structure.

For our English-Chinese air travel information application, we did not have a hand-translated corpus. We built the structures of the English and Chinese monolingual models, a collection of prototypical head machines for various parts of speech, together with some head machines for idiosyncratic words, by hand. We also built prototype local graph structures for bilingual transfer entries, together with idiosyncratic entries for translation idioms in the domain, by hand. This provided a sufficiently constraining model for training with a back-translation process.
3.3 Irregular Events
In our translation system, irregular situations such as missing vocabulary items and fragmentary analysis, transfor-
generation fail outside the model structure. In the costed process framework, we simply treat irregular events
like any other process choices and train cost parameters for them accordingly, thus obviating the need to adjust penalty
values manually. In some experiments, we multiplied the costs of all irregular events by a constant factor greater
than unity. This constant was chosen to prefer regular solutions to irregular ones, but by automatically training the
costs of irregular events, selecting between different irregular solutions is not affected by the constant.

4 COMPUTING EVENT COSTS
4.1 Model Parameter Tables
To manipulate choice costs, we maintain a hash table \( H \), indexed by choice. For any choice \( c \), the associated record in \( H \), \( H(c) \), stores the counts/distances and cost of \( c \).

Any instance of a process induces a list of executed choices, which we call the trace. The above implementation
allows efficient training and subsequent executions. During training, we modify the fields stored in \( H \) appropriately for
each choice contained in a trace. During execution, \( H(c) \) gives the cost of choice \( c \). If there is no such entry in \( H \),
we can either compute a backed off cost for \( c \), or we can use the default cost for unseen events.

Prior to execution, we compute the cost for each choice in \( H \). If we are using backed off costs for unseen events, we
first add to \( H \) entries for the choices with relaxed contexts of those choices already in \( H \). For each costing model
described above, we can then easily compute the cost for all the choices in \( H \) in one or two passes through \( H \).

The above implementation performs the only time-
consuming operation—cost computation—prior to execu-
tion. Cost recording during training and retrieval during
searching reduce to hash-table look-ups. Furthermore, the
implementation facilitates sharing of training data by differ-
ent cost models: the probabilistic and discriminative models
use the same training data, for example.

4.2 Training Procedures
When training, we use two hash tables: the runtime table
and the training table. The runtime table contains the
costs that the system retrieves when executing processes
during training. The training routine itself modifies the
counts/distances that are stored in the training table.

4.2.1 Supervised Training
For supervised training, we first translate a set of sen-
tences. A bilingual speaker then judges each translation
good or bad, and we record the translations and associated
scores. During training, we retranslate all the scored sen-
tences. For each source sentence \( s \), let \( \tau(s) \) be the current
translation of \( s \) into the target language. If \( \tau(s) \)
matches some previously scored translation \( \tau(s) \), then for
each choice \( c \) in the trace of \( \tau(s) \), we increment the posi-
tive (negative) count for \( c \) in the training table if \( \tau(s) \)
was good (bad). This procedure is suitable for the proba-
ibilistic and discriminative models.

4.2.2 Unsupervised Training
We use two unsupervised training procedures. We call the
first method fragmentation training. If a trace for a trans-
faction contains irregular events, we call the translation frag-
mented. During fragmentation training, we translate each
sentence \( s \) into a corresponding \( \tau(s) \). If \( \tau(s) \) is not frag-
mented, then we increment the positive counts in the train-
ing table for the choices in the corresponding trace; oth-
erwise, we increment the negative counts. Furthermore, if
\( \tau(s) \) is not fragmented, we repeat the process in the rever-
se direction, translating \( \tau(s) \) back into the source language.

This training procedure is suitable for the probabilistic and
discriminative models.

The second unsupervised training procedure is suitable
for the distance model. We call it a contrastive method,
because it compares the performance of two models (the
runtime model and a perturbation of the same). Dagan and
Engelson [6] describe the application of such a technique to
probabilistic classification problems.

During contrastive training, each sentence \( s \) in the input
corpus is translated twice, into \( \tau_1(s) \) and \( \tau_2(s) \); \( \tau_1(s) \)
and \( \tau_2(s) \) are then retranslated back to the source language,
producing \( \tau_1(\tau_1(s)) \) and \( \tau_1(\tau_2(s)) \). The translations \( \tau_1(s) \)
and \( \tau_1(\tau_1(s)) \) are computed using the runtime table; the
translations \( \tau_2(s) \) and \( \tau_1(\tau_2(s)) \) are computed using the run-
time table some of the time and randomly chosen costs the
rest of the time. Let \( T_1 \) be the combined traces of \( \tau_1(s) \)
and \( \tau_1(\tau_1(s)) \); similarly define \( T_2 \). Let \( d \) be a string distance
metric; let \( d_1 = d(\tau_1(s), \tau_1(\tau_1(s))) \) and \( d_2 = d(\tau_1(s), \tau_1(\tau_2(s))) \). We
increment the cumulative distances stored in the training
the choices in \( T_1 \) by \( d_1 \); similarly, we increment the
cumulative distances for the choices in \( T_2 \) by \( d_2 \).

The process also increments the occurrence counts for each
choice appropriately. The intuition is that the choices that
both traces contain did not distinguish the divergent trans-
lations, whereas the choices that were contained in only one
trace more directly led to the corresponding translation.

5 EXPERIMENTS AND RESULTS
5.1 ATIS Data
Currently, our ATIS corpus is divided into a training set of
13000 sentences and a test set of 6000 sentences. The entire
training set is used by the unsupervised training procedures.
Approximately 3200 of the training sentences are scored. Of
those, approximately 1150 were matched during the super-
vised training process we report here. (800 matches are
good, and 350 are bad.) Additionally, we have hand tagged
approximately 800 training sentences to derive counts for
prepositional phrase attachment events. We include these
counts with the results of our supervised training procedure
when evaluating our “supervised system.”

5.2 Results
We performed the experiments using all of the training
methods in Section 4.2. For contrastive training, we trans-
slated each sentence five times: once using the runtime table,
and four times each using the runtime table 75% of the time
and randomly chosen costs 25% of the time, producing for
each sentence four contrastive pairings of traces. We ran
contrastive training once using an empty runtime table and
once using a runtime table populated with the supervised
training and hand-tagged data.

We translated 200 test sentences to evaluate the respec-
tive training data. Table 1 shows the results of the eval-
uations. The first column of numbers is percentages of the
test sentences that had good translations, as judged by a
bilingual speaker; the second column represents percent-
ages of the translations that preserved meaning, regardless
of syntax errors. The qualitative numbers reflect the per-
formance of the system with an empty runtime table; i.e.,
all choice costs are default, except that costs of irregular
events are high enough to prefer unfragmented solutions,
and thus the model structure drives the system. The
supervised numbers show the performance using supervised
training, as well as the hand-tagged data and hand-coded
penalty costs. The unsupervised numbers show the per-
formance using contrastive training, in the context-normalized
mean distance model; we also multiplied the costs of the
irregular events by a large enough constant to prefer un-
fragmented solutions. The hybrid numbers show the per-
Table 1. Results of evaluations on various training data.

<table>
<thead>
<tr>
<th>Experiment</th>
<th>Good Translations</th>
<th>Meaning Preserved</th>
</tr>
</thead>
<tbody>
<tr>
<td>Qualitative</td>
<td>29</td>
<td>71</td>
</tr>
<tr>
<td>Supervised</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Probabilistic</td>
<td>46</td>
<td>82</td>
</tr>
<tr>
<td>Discriminative</td>
<td>52</td>
<td>83</td>
</tr>
<tr>
<td>Unsupervised</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Empty runtime</td>
<td>37</td>
<td>71</td>
</tr>
<tr>
<td>Supervised runtime</td>
<td>54</td>
<td>83</td>
</tr>
<tr>
<td>Hybrid</td>
<td>56</td>
<td>83</td>
</tr>
</tbody>
</table>

Figure 1. Statistically significant comparisons of training methods, with respect to good translations. An arc from method \( x \) to method \( y \) means that method \( x \) outperformed method \( y \).

5.3 Interpretation of Data

Figure 1 depicts statistically significant comparisons among the training methods.

While our best results exploit supervised training, the unsupervised contrastive method does improve the performance of the system when bootstrapped from initial supervised training data. Continuing, that the qualitative method performs so well implies that the structures of our models already encode a great deal of knowledge about the language. The differences between the different costing models were much greater with the stricter evaluation test ("good translations" in Table 1); thus the methods affected translation quality more than they did preservation of meaning. This may be partly an artifact of the ATIS domain, in which word sense ambiguity is relatively uncommon. As we move towards automatic learning of the model structure, we expect the differences between training methods to increase.

Finally, we have experimented with iterating the contrastive training method. While we do see some improvement in system performance from iterating, the actual data are not statistically significant.

6 SUMMARY AND CONCLUSION

We have presented a framework and implementation for associating costs with actions taken by a search process; the process choices correspond to equivalence classes of state transitions. Three cost functions were defined, based on log likelihoods, likelihood ratios, and distances in the solution space. Our method treats unseen events systematically.

We then applied costed event models to automatic language translation. Experimental results showed in particular that all three cost models outperformed a baseline model and that the discriminative model outperformed the standard probabilistic model. Using backed off costs for unseen events, however, did not significantly enhance performance.

The distance model provides a new approach to unsupervised training of translation models.

Overall, the implementation of a general event costing mechanism facilitated experimentation with different costing functions, abstracting this from the details of the translation task and the actual search algorithms used. Since the use of costed event models only requires that state transitions of the underlying process are traceable, the framework is readily applicable to problems other than translation.

REFERENCES